

# Tarea Evaluable Docker -

## 2EV\_Ejercicio\_1\_Contenedores en red y Docker Desktop

---

Módulo: Despliegue de aplicaciones Web - Distancia

Alumno: Alfonso Dapena Cores

### Tarea Evaluable Docker - 2EV\_Ejercicio\_1\_Contenedores en red y Docker Desktop

- 1.1 Consideraciones generales
- 1.2 Creación red bridge `redje1`
- 1.3 Creación contenedor con la imagen `mariaDB`
- 1.4 Creación contenedor `phpMyAdmin`.
- 1.5 Conexión de contenedores a la red `redje1`.
- 1.6 Creación script `modulos`, y ejecución del mismo en nuestra base de datos `DAW`
- 1.7 Acceso a la base de datos.
- 1.8 Instalamos extensión `Resource usage`

### 1.1 Consideraciones generales

- ¿Qué es Docker?

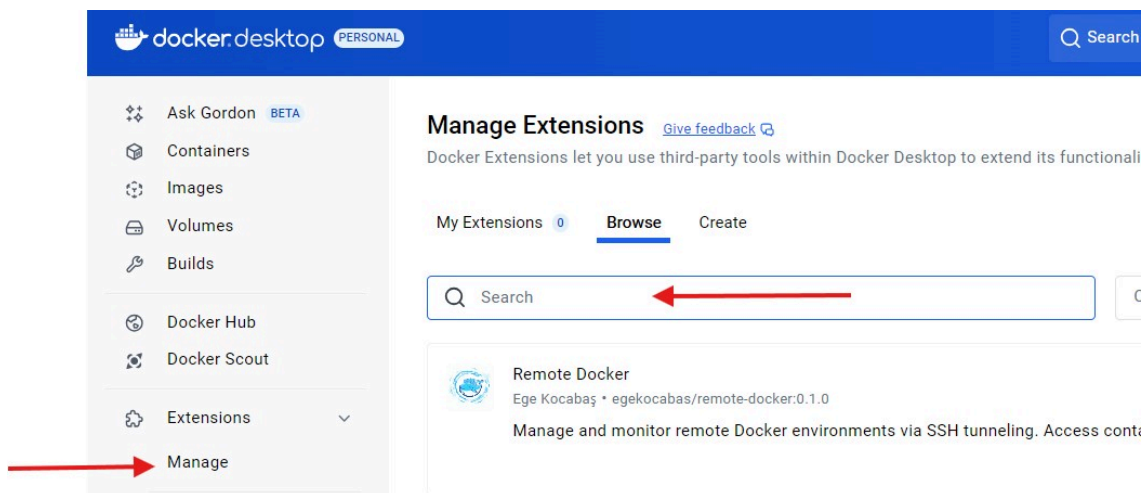
Docker es una plataforma de código abierto que permite crear, ejecutar y gestionar aplicaciones en contenedores. Los contenedores son unidades estandarizadas que combinan el código de la aplicación con sus dependencias. De manera similar a cómo una máquina virtual virtualiza el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

Con Docker, podemos utilizar los contenedores como máquinas virtuales livianas y modulares, y obtener la flexibilidad necesaria para crearlos, implementarlos, copiarlos y trasladarlos de un entorno a otro. Docker permite empaquetar y ejecutar una aplicación en un entorno ligeramente aislado llamado contenedor. El aislamiento y la seguridad permiten ejecutar varios contenedores simultáneamente en un host. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, por lo que no es necesario depender de lo que esté instalado en el host. Podemos compartir contenedores mientras trabajamos y nos aseguramos de que todos los usuarios con los que lo compartimos obtengan el mismo contenedor, que funciona de la misma manera.

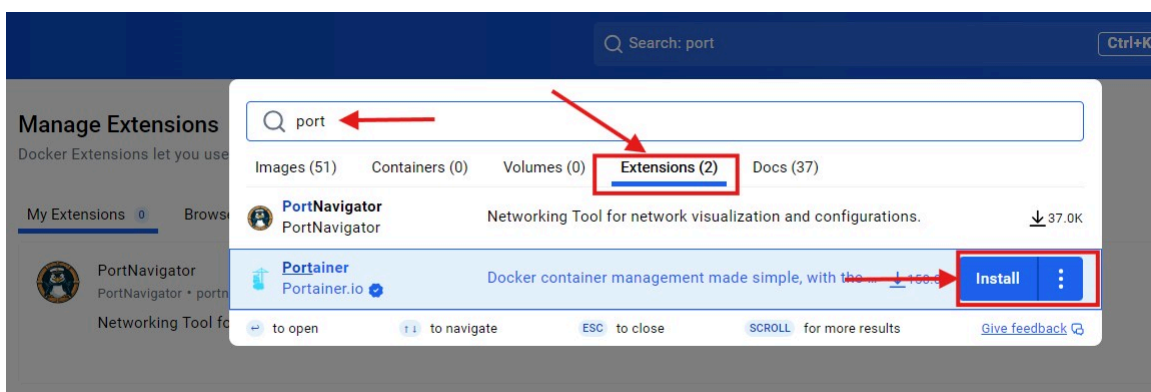
Docker proporciona herramientas y una plataforma para administrar el ciclo de vida de sus contenedores, denominada Docker Hub, que es un repositorio público de imágenes de contenedores que permite almacenar, administrar y compartir imágenes de Docker, que son las plantillas que definen un contenedor, es decir, el espacio para ejecutar aplicaciones y servicios. Las imágenes Docker contienen el código, las bibliotecas y dependencias necesarias para que nuestro contenedor funcione.

## 1.2 Creación red bridge redge1

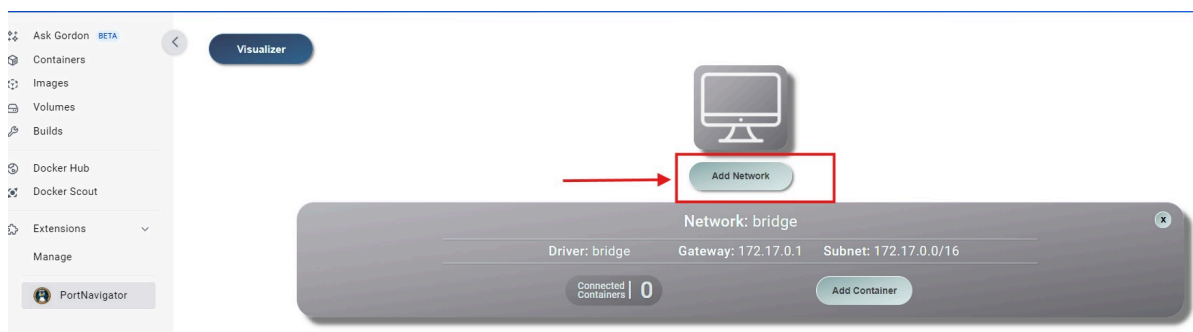
- En primer lugar, accedemos a la aplicación Docker Desktop, pinchamos en la pestaña **Extensions** y desplegamos la opción **Manage**:



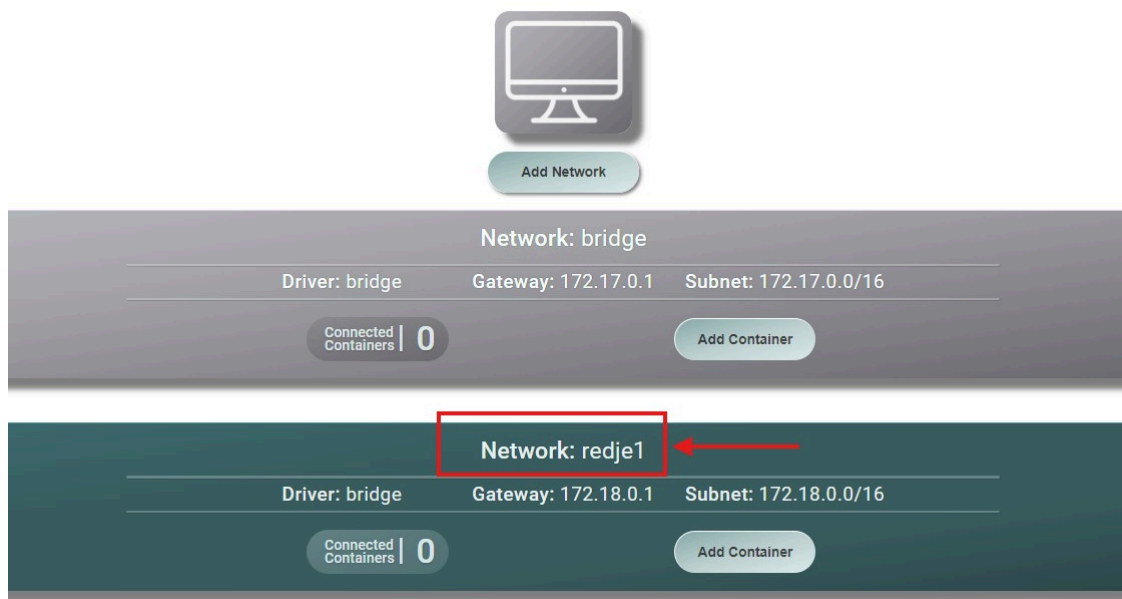
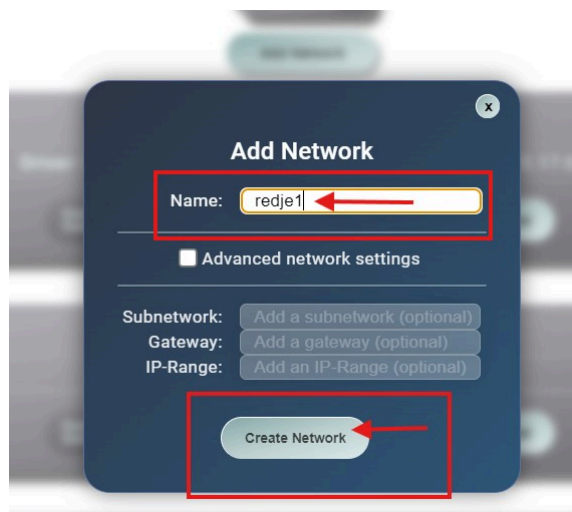
- Buscamos la extensión **PortNavigator** y la descargamos:



- A continuación, crearemos la red bridge **redge1** para acceder a la pestaña de la extensión **PortNavigator**, y pinchamos en la opción **Add network**:

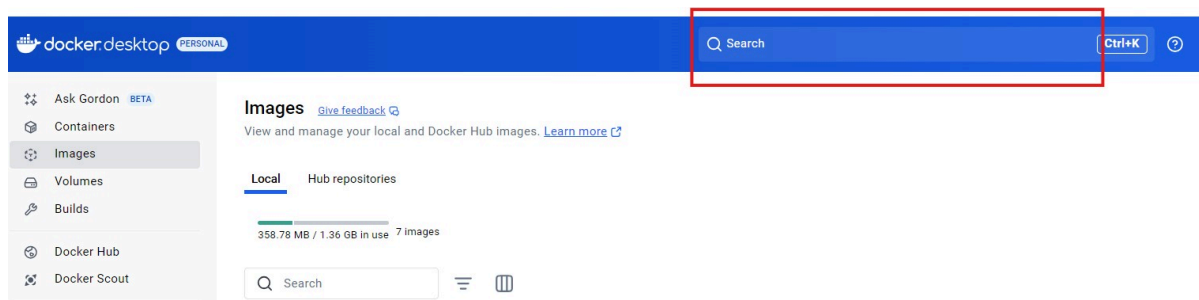


- Le damos el nombre a la red y la creamos. Por defecto el propio programa asignará las direcciones IP a nuestra red.

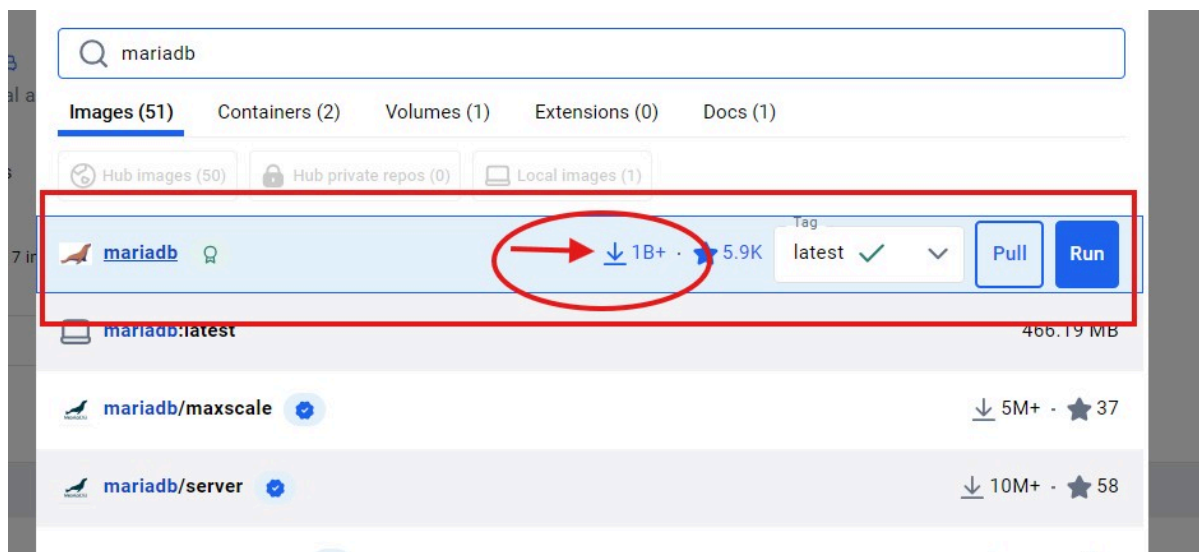


### 1.3 Creación contenedor con la imagen mariadb

- En primer lugar buscaremos en el navegador de Docker Desktop la imagen mariadb :



- A continuación, examinaremos nuestra imagen. Este apartado nos mostrará información relativa a la imagen: uso, cómo crear un contenedor a partir de ella, las versiones de la misma, como crear un fichero Dockerfile. Nosotros vamos a prestar especial atención a la sección Environment variables, para configurar las variables de entorno con las que definiremos la contraseña y el usuario de nuestra base de datos:



mariadb

Tag  
latest ✓

Docker Official Image
 1B+ · 5.9K
 [View on Hub](#)

```
docker run mariadb
```

Note: Additional parameters might be required. See full image description below to learn more.

MariaDB Server is a high performing open source relational database, forked from MySQL.

## Quick reference

- Maintained by:  
[MariaDB Foundation](#), [MariaDB plc](#), with contributions from our [community](#)
- Where to get help:  
[Database Administrators \(Stack Exchange\)](#), [MariaDB Knowledge Base \(Ask a Question here\)](#).

Also see the ["Getting Help with MariaDB" article on the MariaDB Knowledge Base](#).

## Supported tags and respective Dockerfile links

- [11.8.1-ubi9-rc](#) [11.8-ubi9-rc](#) [11.8.1-ubi-rc](#) [11.8-ubi-rc](#)
- [11.8.1-noble-rc](#) [11.8-noble-rc](#) [11.8.1-rc](#) [11.8-rc](#)
- [11.7.2-ubi9](#) [11.7-ubi9](#) [11-ubi9](#) [11.7.2-ubi](#) [11.7-ubi](#) [11-ubi](#)
- [11.7.2-noble](#) [11.7-noble](#) [11-noble](#) [noble](#) [11.7.2](#) [11.7](#) [11](#) [latest](#)

mariadb

Tag  
latest ✓

Docker Official Image
 1B+ · 5.9K
 [View on Hub](#)

Updated 9 days ago

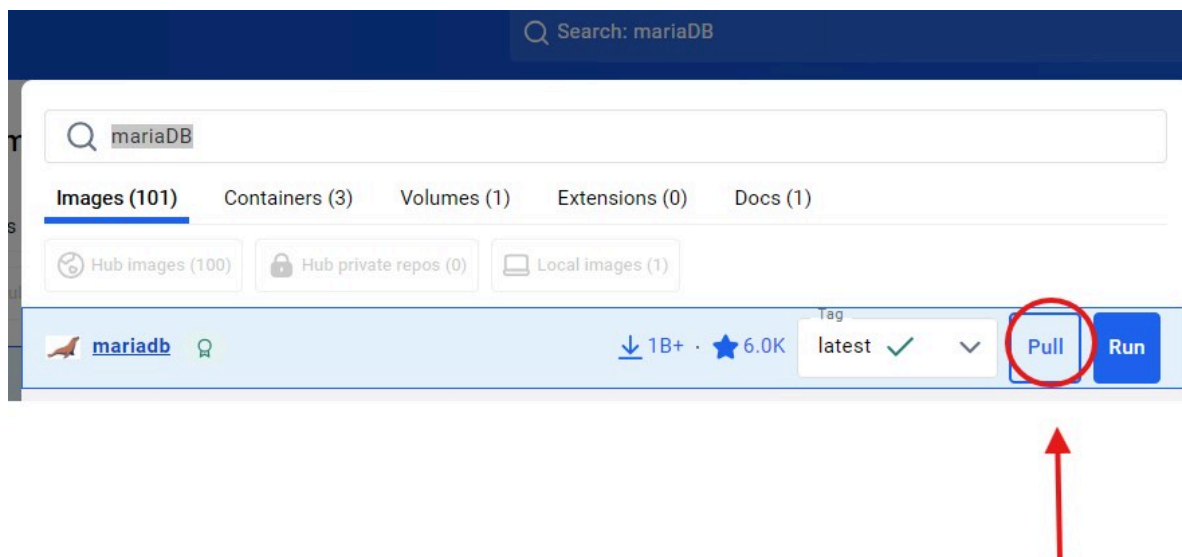
Environment Variables

When you start the `mariadb` image, you can adjust the initialization of the MariaDB instance by passing one or more environment variables on the `docker run` command line. Do note that all of the variables, except `MARIADB_AUTO_UPGRADE`, will have no effect if you start the container with a data directory that already contains a database. I.e. any pre-existing database will always be left untouched on container startup.

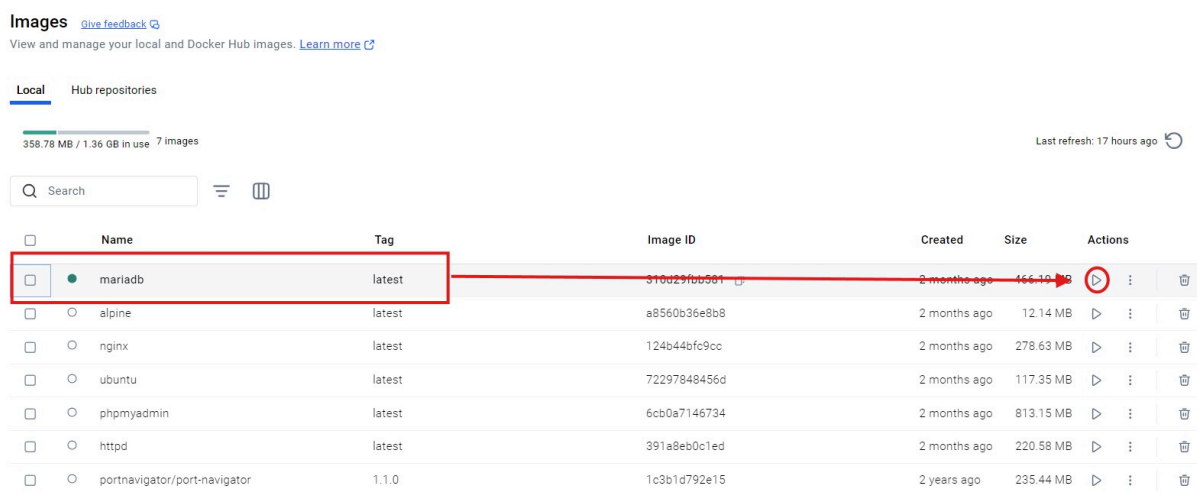
One of `MARIADB_RANDOM_ROOT_PASSWORD`, `MARIADB_ROOT_PASSWORD_HASH`, `MARIADB_ROOT_PASSWORD` or `MARIADB_ALLOW_EMPTY_ROOT_PASSWORD` (or equivalents, including `*_FILE`), is required. The other environment variables are optional.

There is a large list of environment variables and the complete list is documented on [MariaDB's Knowledge Base : MariaDB Server Docker Official Image Environment Variables](#).

- A continuación, descargamos la imagen con la opción `pull`:



- Una vez hemos descargado nuestra imagen, procedemos a crear el contenedor desde la sección de imágenes y pinchando en el botón de `play`, situado debajo de `Actions`:



- A continuación, se nos abrirá una pestaña donde indicaremos el nombre de nuestro contenedor `mariadb`, el puerto 3306 a través del que nuestro contenedor será accesible. En la opción `volumes` crearemos el volumen `ejercicio1` en el directorio `/var/lib/mysql`, para que el almacenamiento de nuestra base de datos sea persistente. En la pestaña `Environment variables`, introduciremos nuestra contraseña root `MYSQL_ROOT_PASSWORD`, 1234, el nombre de la base de datos por defecto `daw`, un usuario `MYSQL_USER`, "alfonsodapena", con su contraseña `MYSQL_PASSWORD`, "alfonso". Después de introducir todos los datos, pulsamos `RUN` para crear nuestro contenedor, que se inicializará de inmediato.



## Run a new container

mariadb:latest

### Optional settings

Container name

Contenedor\_mariaDB

A random name is generated if you do not provide one.

### Ports

Enter "0" to assign randomly generated host ports.

Host port

3306

:3306/tcp

### Volumes

Host path

ejercicio1

Container path

/var/lib/mysql

### Environment variables

Variable

MYSQL\_ROOT\_PASSWORD

Value

1234

Variable

MYSQL\_DATABASE

Value

daw

Variable

MYSQL\_USER

Value

alfonsodapena

Variable

MYSQL\_PASSWORD

Value

alfonso

Cancel

Run

## Containers

View all your running containers and applications. [Learn more](#)

Container CPU usage

0.03% / 1200% (12 CPUs available)

Container memory usage

325.2MB / 7.44GB

Search



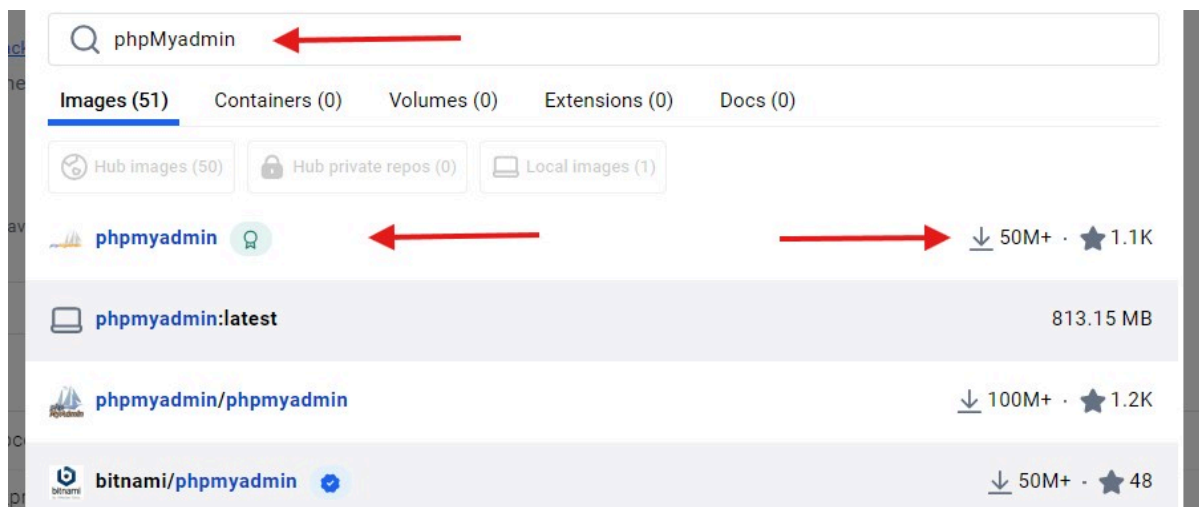
Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	<input type="radio"/> mariadbcontainer	e02aac9c3a41	<a href="#">mariadb:latest</a>		0%	15 hours ago	
<input type="checkbox"/>	<input type="radio"/> prueba_practica_3	e9b805d60313	<a href="#">mariadb:latest</a>		0%	17 hours ago	
<input checked="" type="checkbox"/>	<input checked="" type="radio"/> Contenedor_mariaDB	6aec5f514111	<a href="#">mariadb:latest</a>	<a href="#">3306:3306</a>	0.03%	41 seconds ago	

## 1.4 Creación contenedor phpMyAdmin.

- Para la creación de nuestro contenedor `contenedor_phpmyadmin`, seguiremos los pasos indicados en la creación del contenedor `Contenedor_mariaDB`, del punto 1.2:
  - Buscamos la imagen en el navegador.
  - Examinamos la imagen.
  - Bajamos la imagen y creamos el contenedor, con sus variables de entorno, donde indicaremos la variable de entorno `PMA_HOST`, que especifica el `host MySQL` para `phpMyAdmin`.



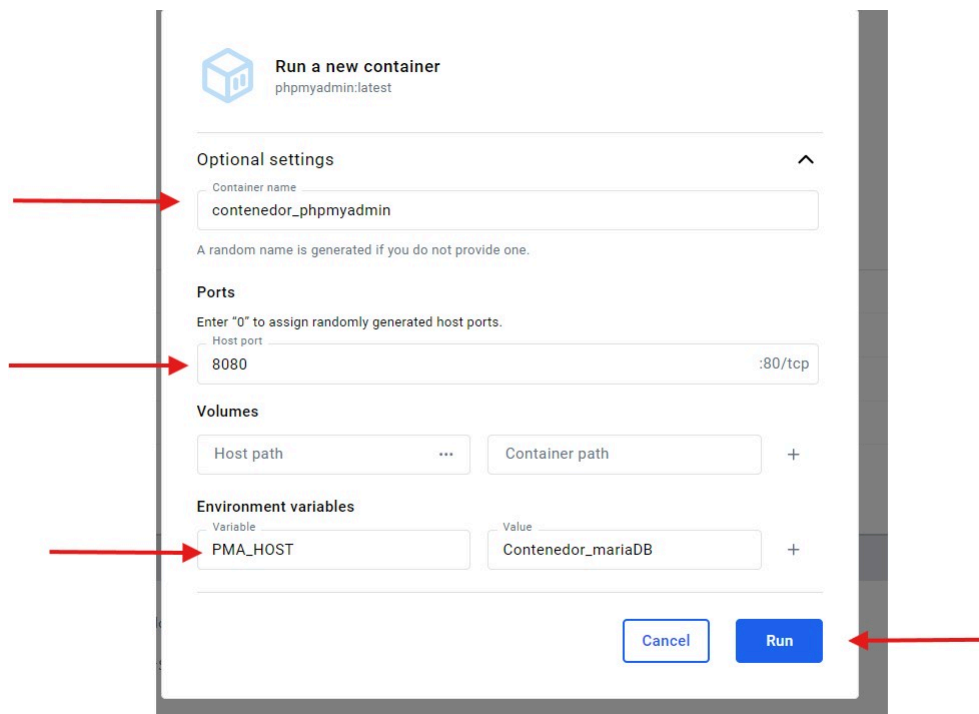


Local Hub repositories

358.78 MB / 1.36 GB in use 7 Images Last refresh: 17 hours ago

Search

	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	● mariadb	latest	310d29fbb581	2 months ago	466.19 MB	▶ : 🗑
<input type="checkbox"/>	○ alpine	latest	a8560b36e8b8	2 months ago	12.14 MB	▶ : 🗑
<input type="checkbox"/>	○ nginx	latest	124b44bfc9cc	2 months ago	278.63 MB	▶ : 🗑
<input type="checkbox"/>	○ ubuntu	latest	72297848456d	2 months ago	117.35 MB	▶ : 🗑
<input type="checkbox"/>	○ phpmyadmin	latest	6eb0e7146724	2 months ago	813.15 MB	▶ : 🗑
<input type="checkbox"/>	○ httpd	latest	391a8eb0c1ed	2 months ago	220.58 MB	▶ : 🗑
<input type="checkbox"/>	○ portnavigador/port-navigador	1.1.0	1c3b1d792e15	2 years ago	235.44 MB	▶ : 🗑

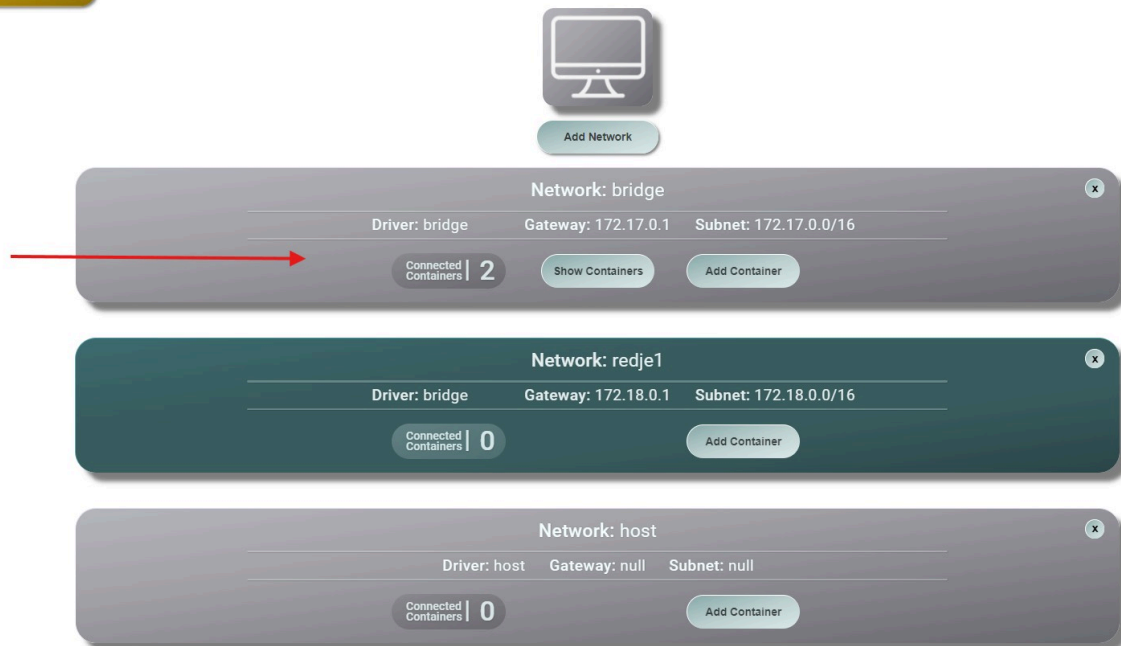


## 1.5 Conexión de contenedores a la red `redje1`.

- El siguiente paso será conectar a los dos contenedores a la misma red, en este caso la `redje1`. Por defecto `Docker Desktop`, al crear los contenedores los asigna a un red, por lo que antes deberemos de desconectarlos de la red a la que han sido asignados los contenedores automáticamente:

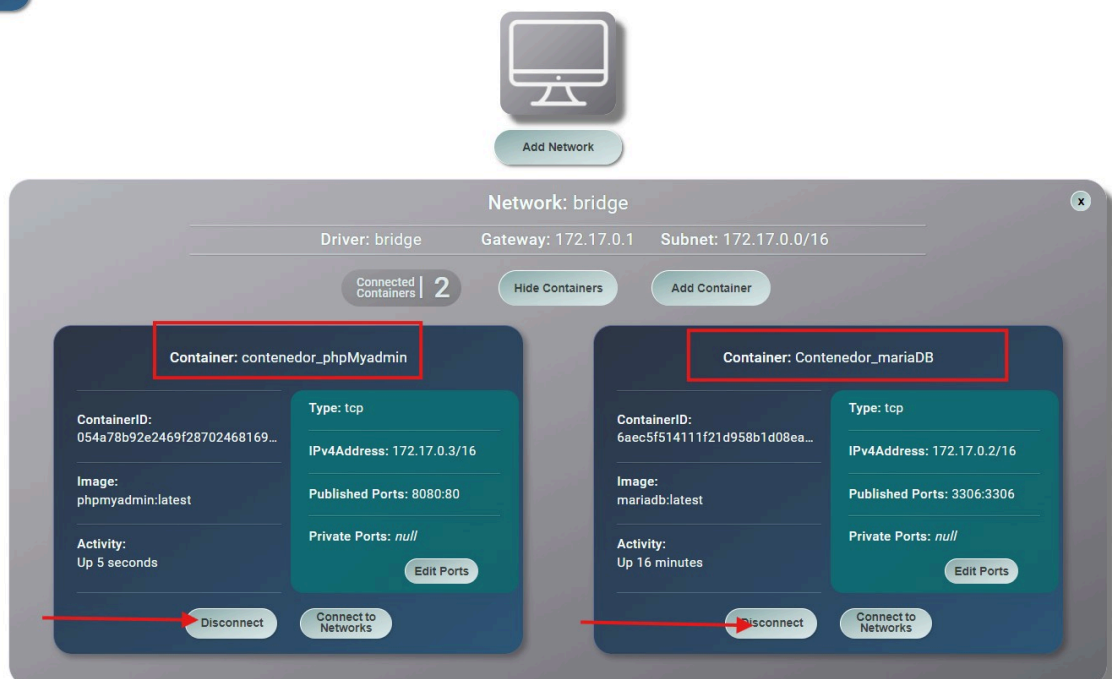
### 1. Conexión por defecto:

Visualizer



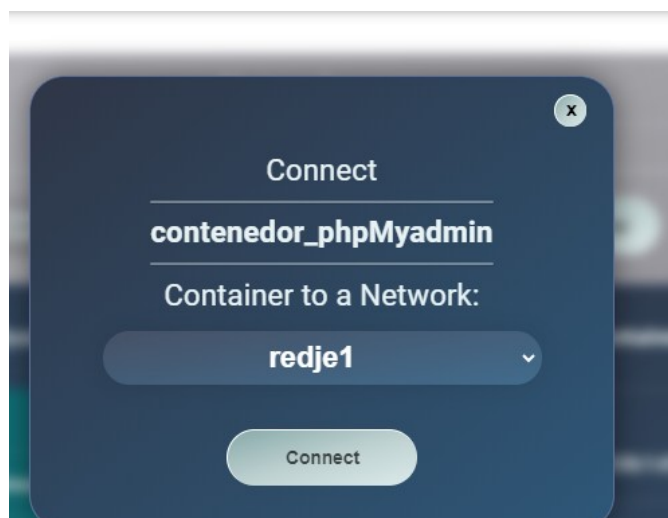
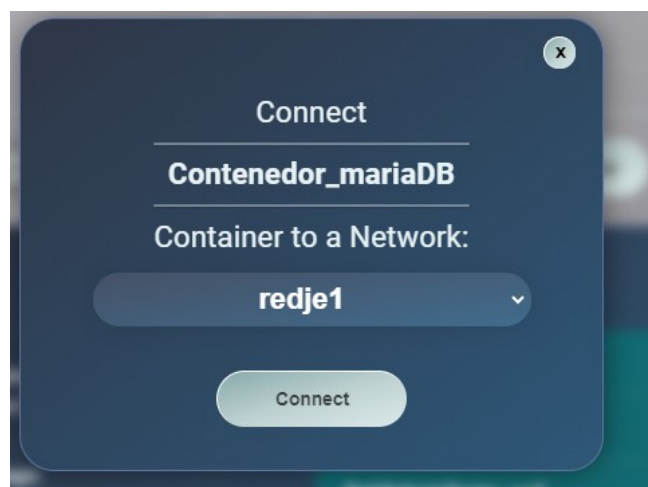
2. Procedemos a desconectarlos de la red

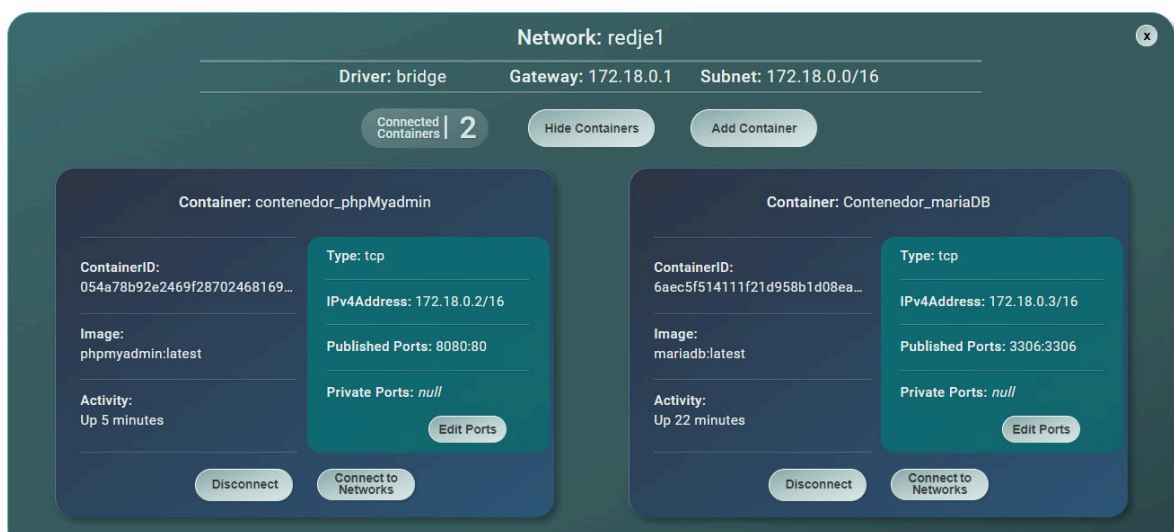
Visualizer



3. Después de desconectarlos, nos fijamos en que en la parte de abajo, figuran ya los dos contenedores desconectados por lo que ya podemos proceder a conectarlos a la red `redje1`







## 1.6 Creación script `modulos` , y ejecución del mismo en nuestra base de datos `DAW`

- En primer lugar debemos de crear un script SQL, que cree una tabla `modulos` . Una vez creado este archivo, copiaremos el mismo al fichero raíz del `Contenedor_mariaDB` creado. Para ello abrimos la terminal integrada de `Docker Desktop` , y copiamos nuestro archivo `modulos.sql` al directorio `/var/lib/mysql`:

```
$ docker cp C:\Users\adcor\modulos.sql Contenedor_mariaDB:/var/lib/mysql
```

**Containers** [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage ⓘ 0.01% / 1200% (12 CPUs available) Container memory usage ⓘ 394.28MB / 7.44GB [Show charts](#)

Search  ☐ Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Actions
<input type="checkbox"/>	contenedor_php	360b51274c5f	phpmyadm	8080:80	0%	<input type="checkbox"/> <input type="vertical-ellipsis"/> <input type="trash"/>
<input type="checkbox"/>	Contenedor_ma	6aec5f514111	mariadb:lat	3306:3306	0.02%	<input type="checkbox"/> <input type="vertical-ellipsis"/> <input type="trash"/>
<input type="checkbox"/>	mariadbcontain	e02aac9c3a41	mariadb:lat		0%	<input type="checkbox"/> <input type="vertical-ellipsis"/> <input type="trash"/>
<input type="checkbox"/>	prueba_practica	e9b805d60313	mariadb:lat		0%	<input type="checkbox"/> <input type="vertical-ellipsis"/> <input type="trash"/>

Showing 4 items

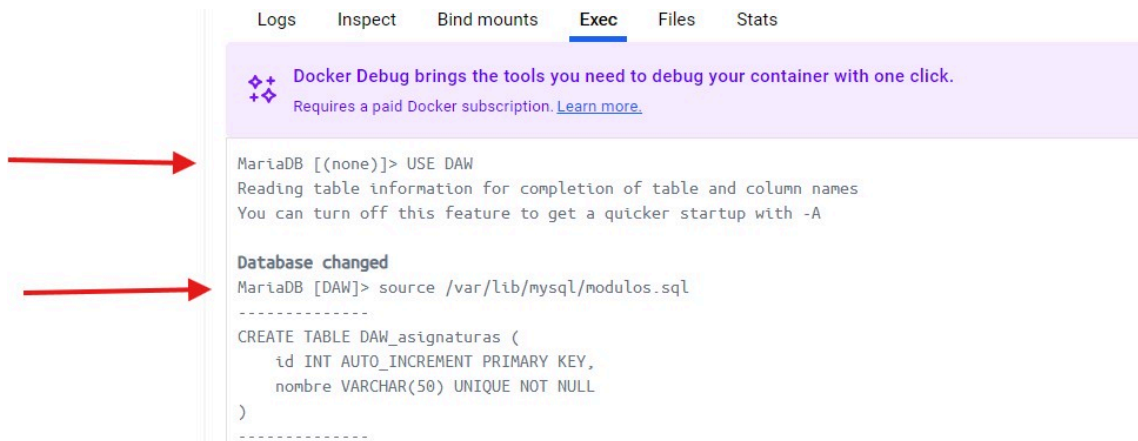
RAM 1.30 GB CPU 0.08% Disk: 2.92 GB used (limit 1006.85 GB)

[Terminal](#) New version available



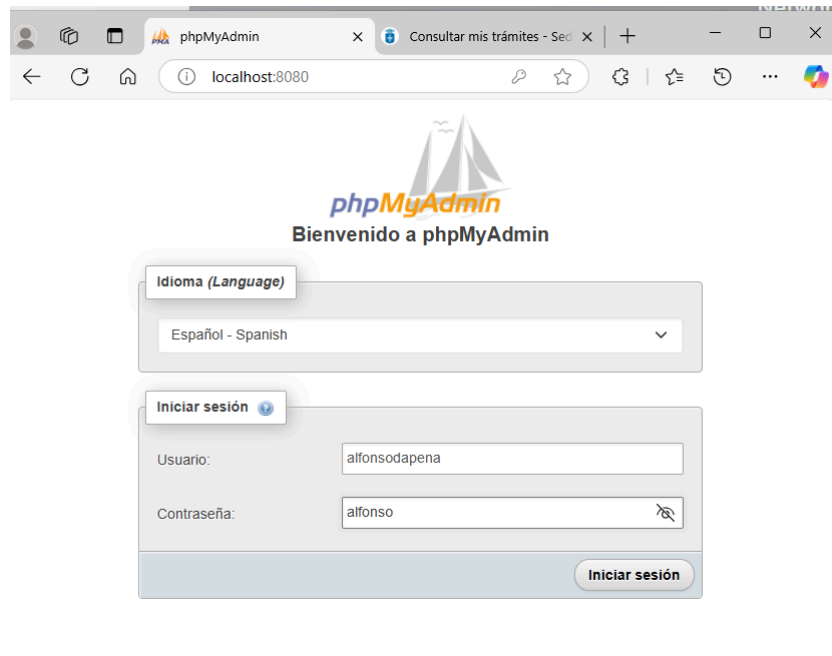
- A continuación, cambiamos a la base de datos `DAW` y ejecuto el script `modulos.sql`, para crear la tabla `modulos`. Para ello introducimos las siguientes instrucciones:

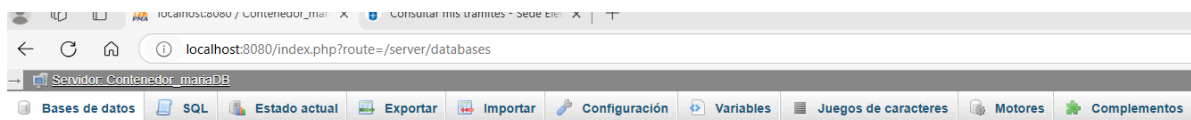
```
USE DAW
source /var/lib/mysql/modulos.sql
```



## 1.7 Acceso a la base de datos.

- Para comprobar que nuestra base de datos funciona correctamente, accedemos a nuestro navegador indicando `localhost:8080`, para acceder a la aplicación `phpMyAdmin`, que nos mostrará la tabla `modulos`.





## Bases de datos

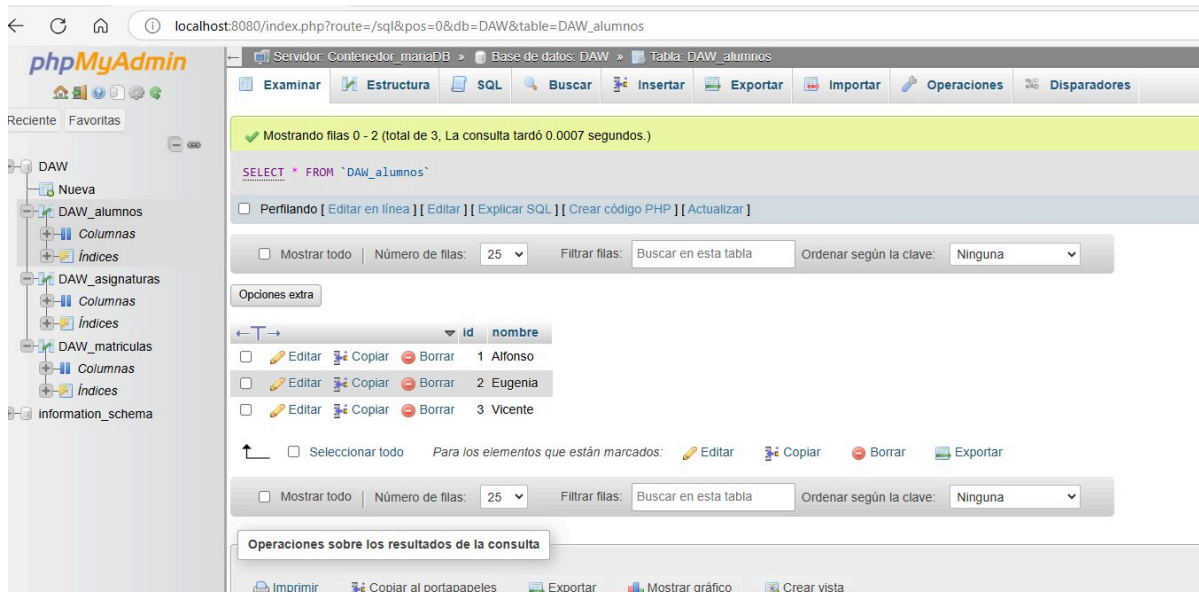
Crear base de datos

No tienes permisos para crear bases de datos

Base de datos	Cotejamiento	Acción
DAW	utf8mb4_uca1400_ai_ci	Seleccionar privilegios
information_schema	utf8mb3_general_ci	Seleccionar privilegios
Total: 2		

Nota: Activar aquí las estadísticas de la base de datos podría causar tráfico pesado entre el servidor web y el servidor MySQL.

Activar estadísticas

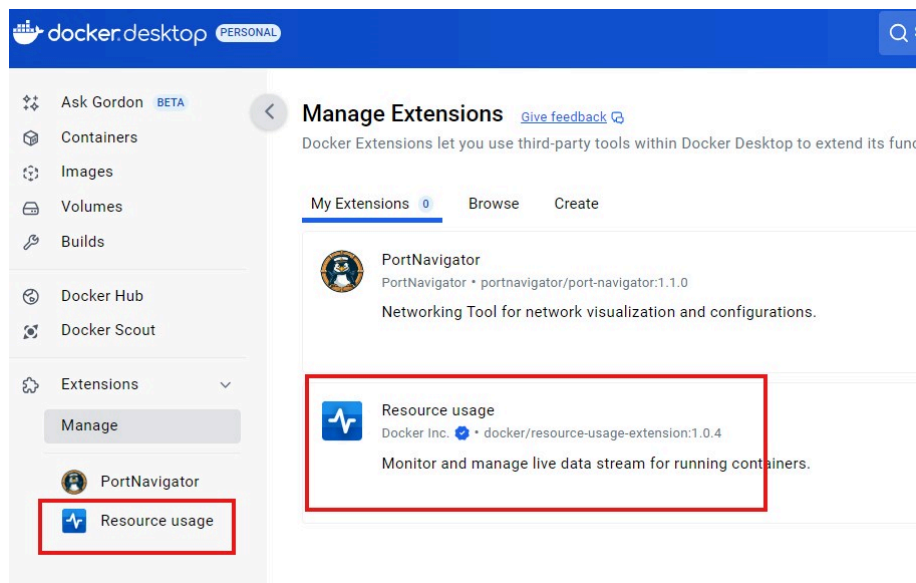
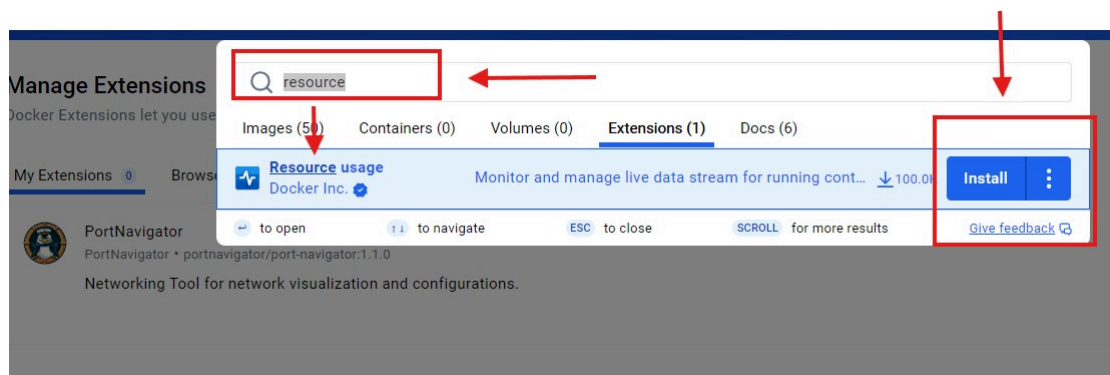






## 1.8 Instalamos extensión Resource usage

- Del mismo modo que instalamos la extensión PortNavigator, accedemos a la pestaña de búsqueda en la aplicación, realizamos su búsqueda y la instalamos:



- Una vez instalada, arrancamos nuestros contenedores y comprobamos nuestra salida cuando estos están activos. La extensión permite dos vistas: Table view, y Chart view.



Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Manage

PortNavigator

Resource usage

Resource Usage [Give Feedback](#)

Monitor and manage live data stream for running containers.

Last updated on: 17/4/2025, 19:18:48 Refresh rate 5s

CPU CORES USAGE 0

Allocated: 12 0.02% / 1200%

MEMORY USAGE

384.84MB / 7.62GB

CONTAINERS

Application containers: 4 2 / 4 running

System containers: 0

[Go to containers](#)

Table view Chart view

Columns Filters

Name	Status	CPU (%)	Memory Usage/LI...	MEM (%)	Disk Read/Write	Network I/O	PIDS	
Contenedor_mariaDB	running	0.01%	331.7MB / 7.62GB	4.25%	0B / 0B	23.8KB / 86.5KB	10	<div></div> ⋮
contenedor_phpmyadmin	running	0.01%	53.14MB / 7.62GB	0.68%	0B / 0B	104KB / 74.3KB	10	<div></div> ⋮
mariaadbcontainer	not running	0.00%	0B / 0B	0.00%	0B / 0B	0B / 0B	0	<div></div> ⋮
prueba_practica_3	not running	0.00%	0B / 0B	0.00%	0B / 0B	0B / 0B	0	<div></div> ⋮

Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Manage

PortNavigator

Resource usage

Resource Usage [Give Feedback](#)

Monitor and manage live data stream for running containers.

Last updated on: 17/4/2025, 19:19:43 Refresh rate 5s

CPU CORES USAGE 0

Allocated: 12 0.05% / 1200%

MEMORY USAGE

377.73MB / 7.62GB

CONTAINERS

Application containers: 4 2 / 4 running

System containers: 0

[Go to containers](#)

Table view Chart view

CPU usage

Memory usage